Efficient Point-Based Trajectory Search

Shuyao Qi, The University of Hong Kong

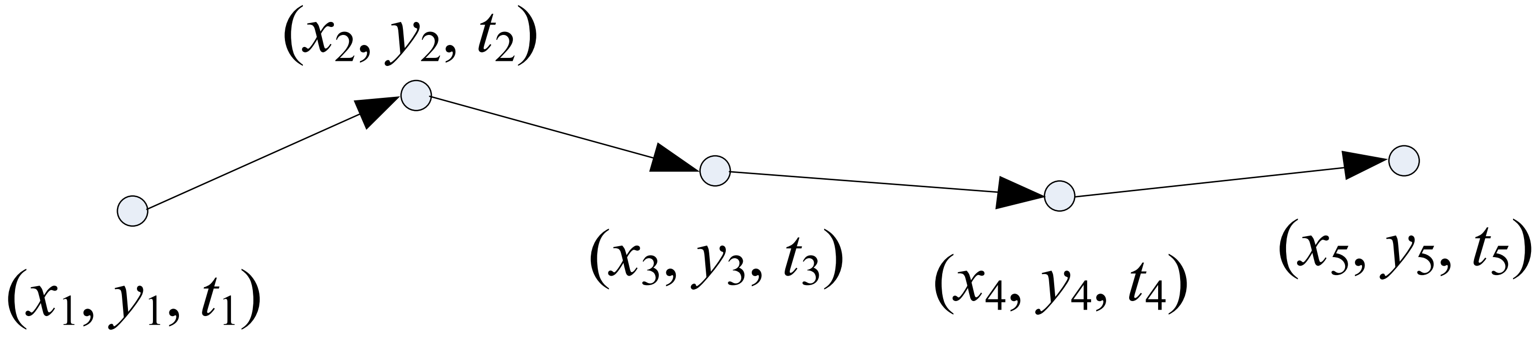
Panagiotis Bouros, Humboldt-Universität zu Berlin

Dimitris Sacharidis, Technische Universität Wien

Nikos Mamoulis, The University of Hong Kong

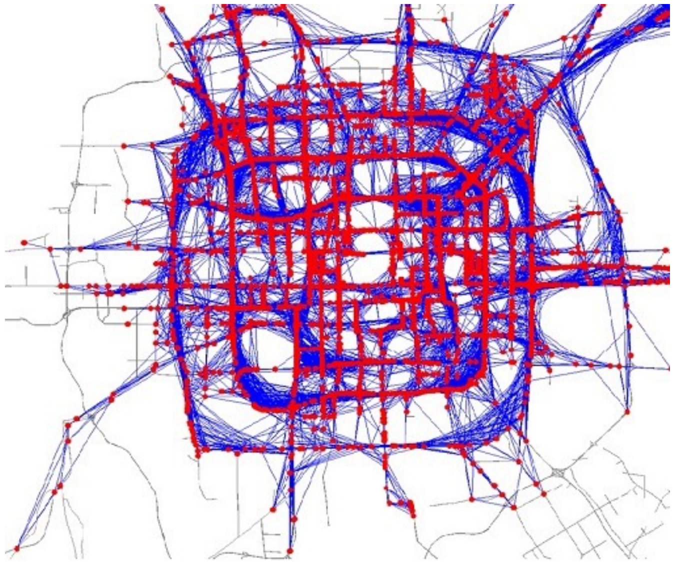
Trajectory Data

- A trajectory is a sequence of **spatio-temporal records**, indicating the travelling history of an object



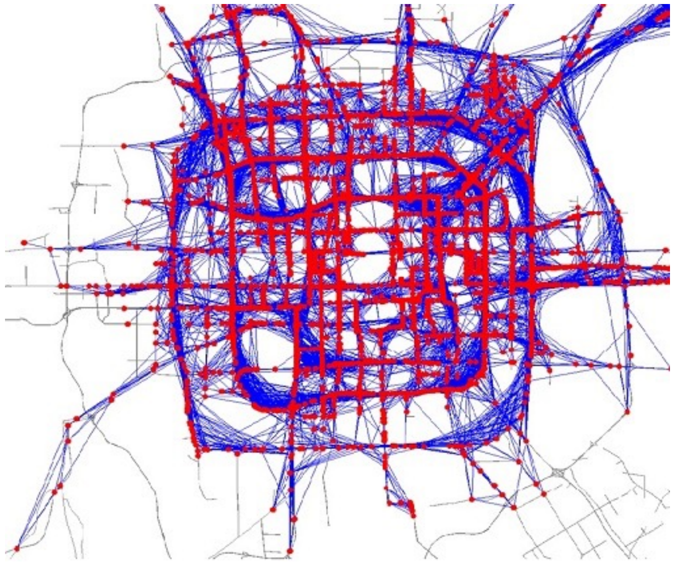
Trajectory Search

- Given a database D of trajectories, find the subset of trajectories qualifying a given criterion
  - P-query: search trajectories w.r.t. given point(s)
  - R-query: search trajectories w.r.t. a given range
  - T-query: search trajectories w.r.t. a selected trajectory



Trajectory Search

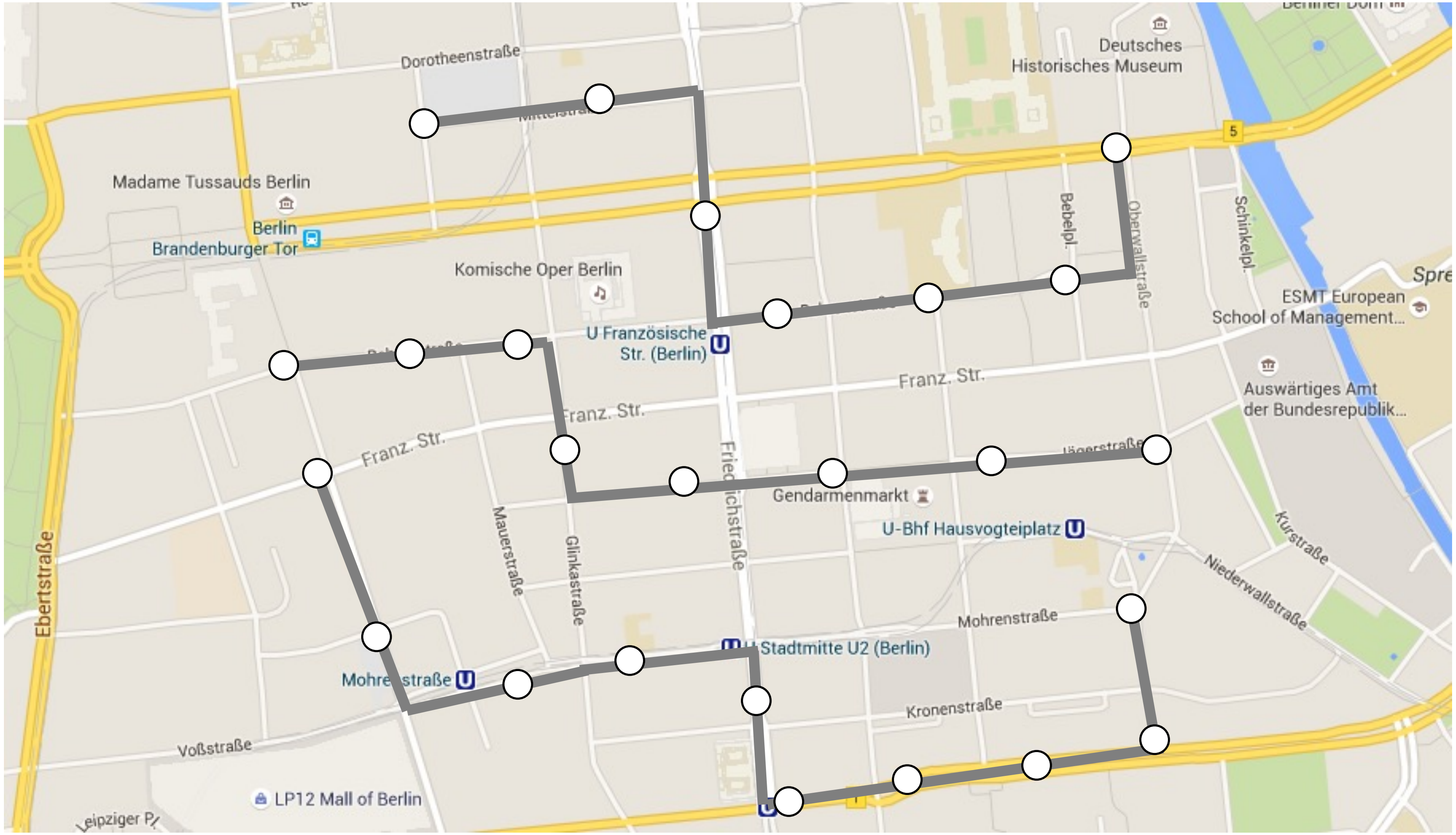
- Given a database D of trajectories, find the subset of trajectories qualifying a given criterion
  - **P-query: search trajectories w.r.t. given point(s)**
  - R-query: search trajectories w.r.t. a given range
  - T-query: search trajectories w.r.t. a selected trajectory



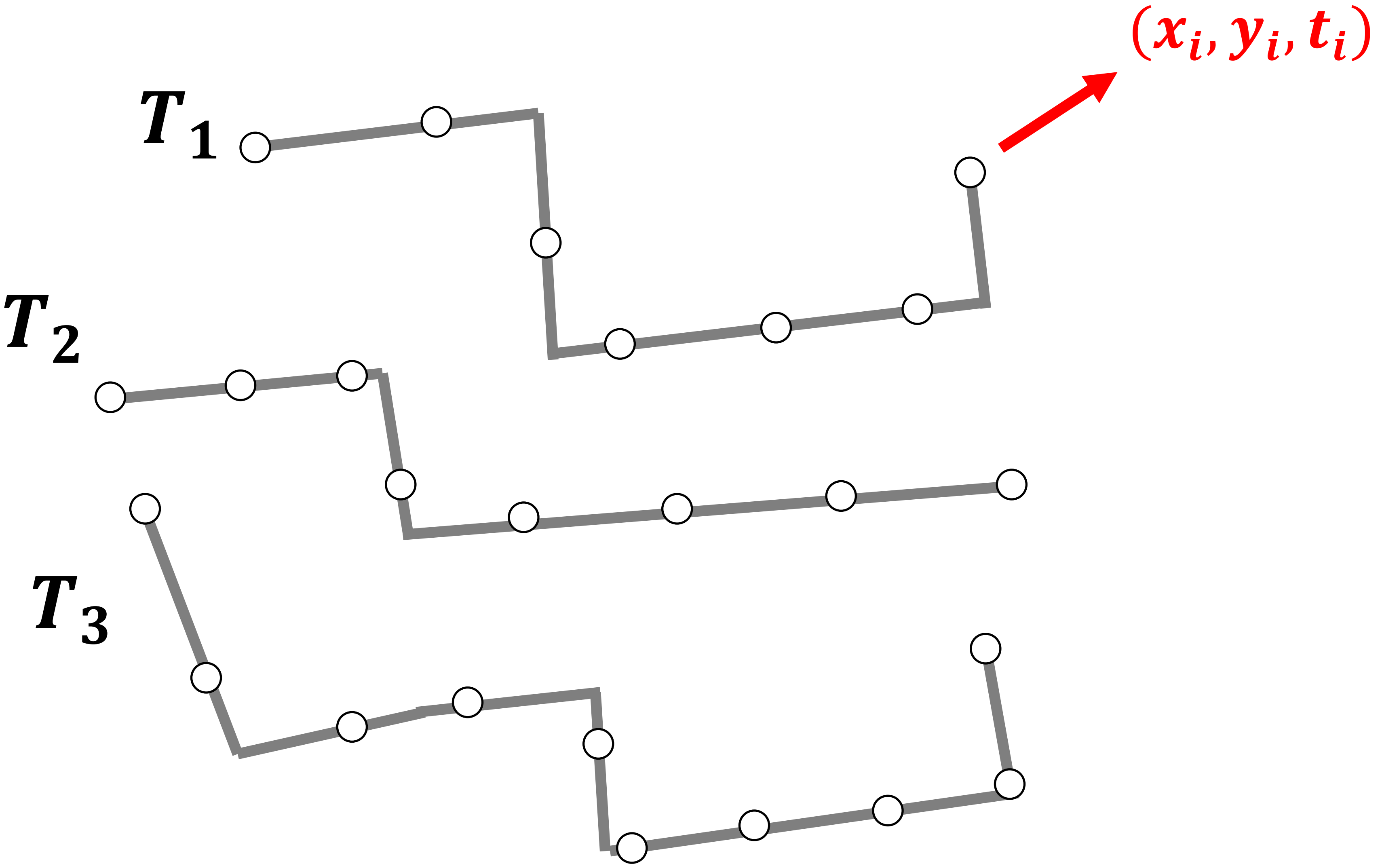
Outline

- Distance-Based Trajectory Search
  - Existing solutions and their drawbacks
  - SRA and SRA+: Novel and efficient approaches
- Bounded Distance-Based Trajectory Search
- Other variants
- Experiments
- Conclusion

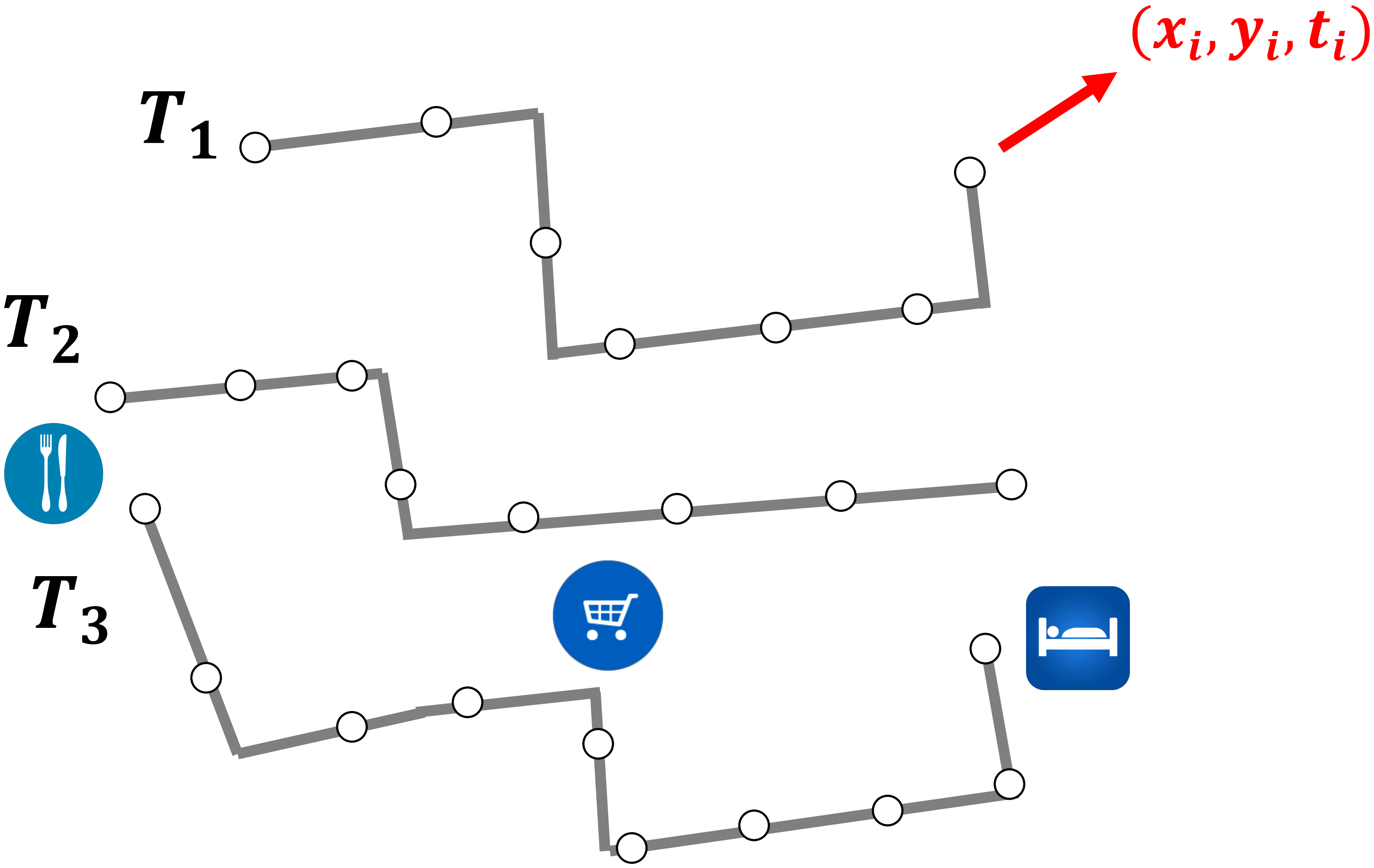
Distance-based Trajectory Search (*k*-DTS)



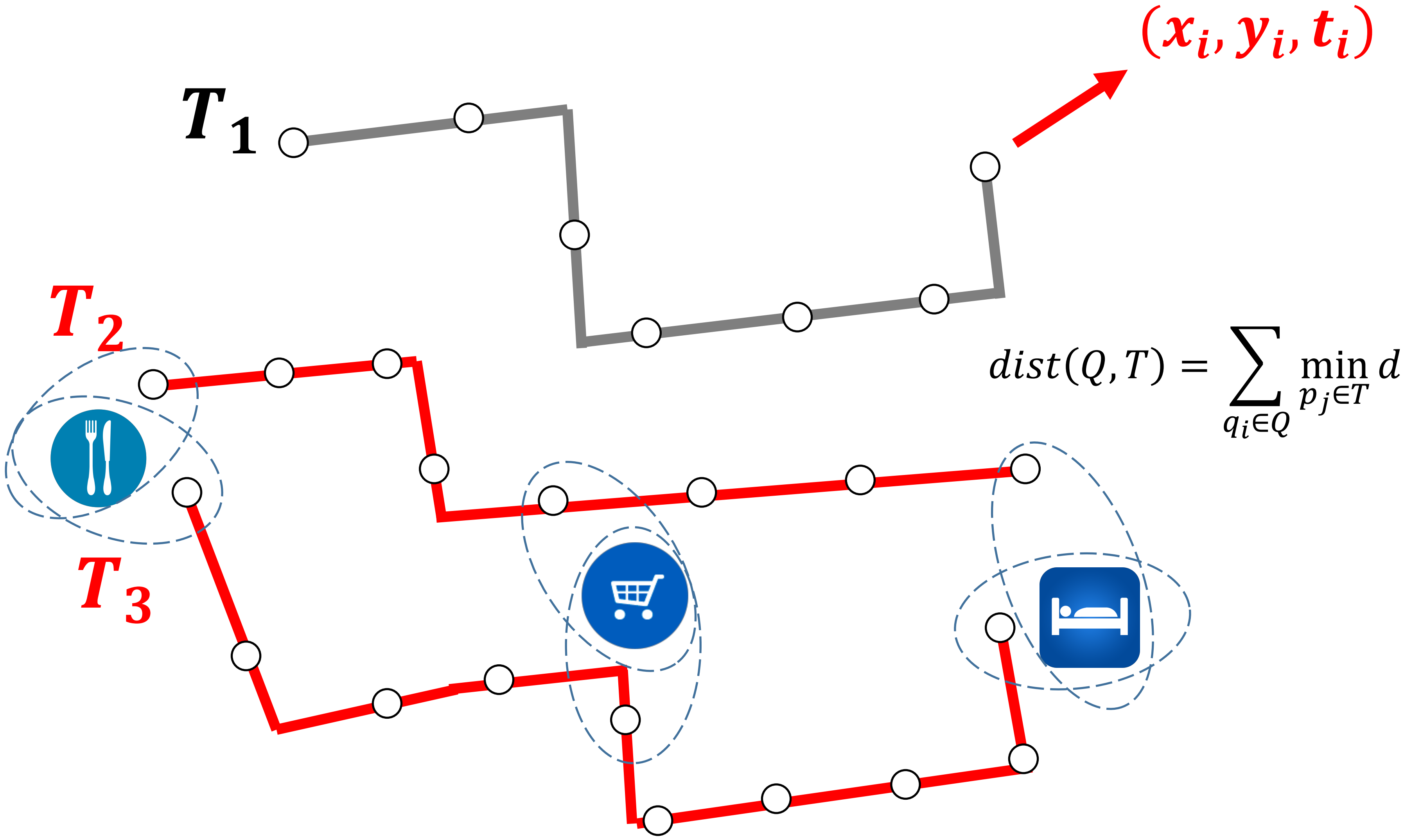
Distance-based Trajectory Search (k-DTS)



Distance-based Trajectory Search (*k*-DTS)



Distance-based Trajectory Search (*k*-DTS)



$$dist(Q,T) = \sum_{q_i \in Q} \min_{p_j \in T} dist(q_i, p_j)$$

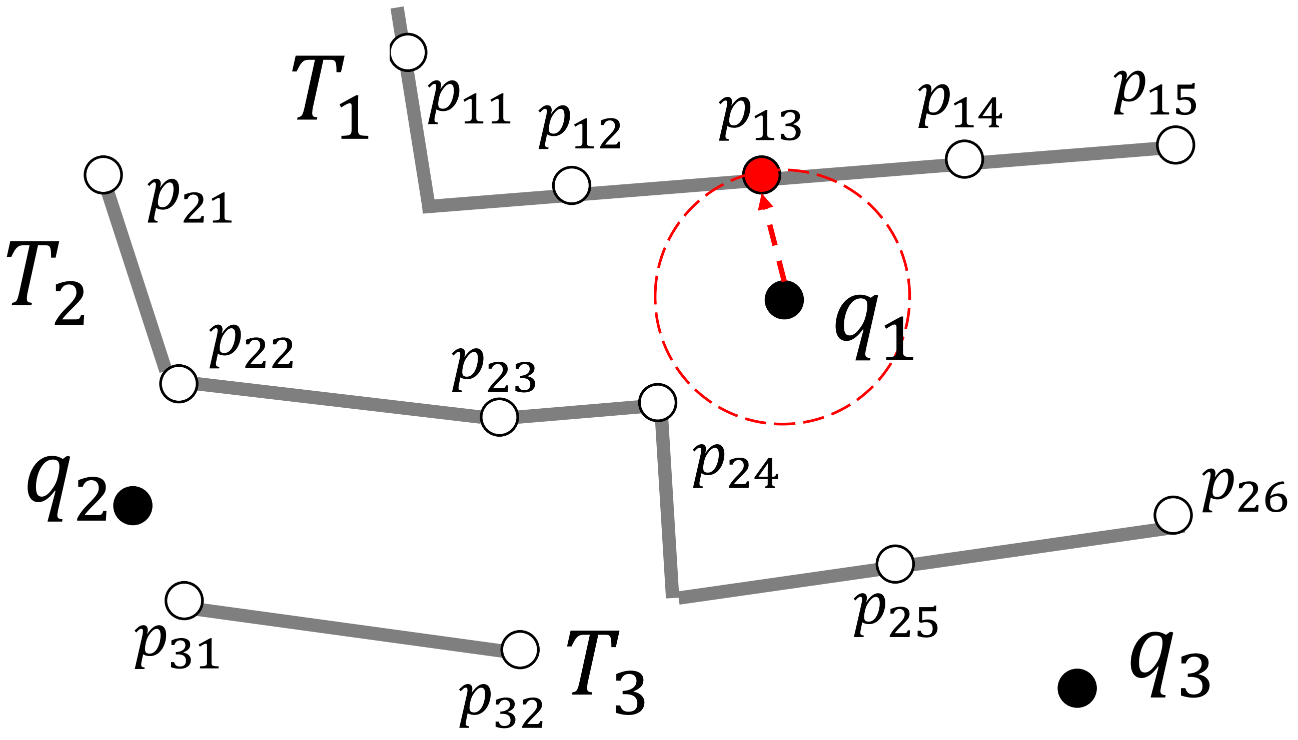
Distance-based Trajectory Search (k-DTS)

- [Chen 2010, Tang 2011]
- Given a trajectory database T
- Input a set of locations $Q = \{q_1, q_2, \ldots, q_n\}$
- Find the top-k trajectories that are closest to Q
  - Formally: $dist(Q, T) = \sum_{q_i \in Q} \min_{p_j \in T} dist(q_i, p_j)$
- Route recommendation, trajectory mining, traffic study, etc.
- Unordered / Ordered

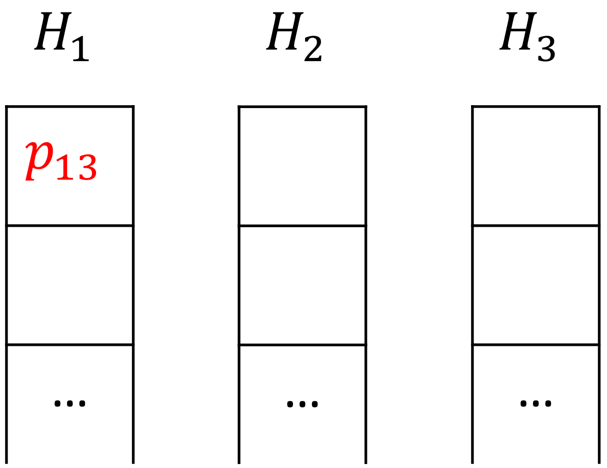
Existing Approaches: NN-based Solutions

- IKNN [Chen 2010], GH/QE [Tang 2011]
- Given trajectory database T, index all trajectory points by an **R-tree**
- **Generation-Refinement** Framework
  - Step1: **Generate** trajectories candidates using NN queries
  - Step2: **Refine** the candidates to find the top-*k* results

Generation with NN queries



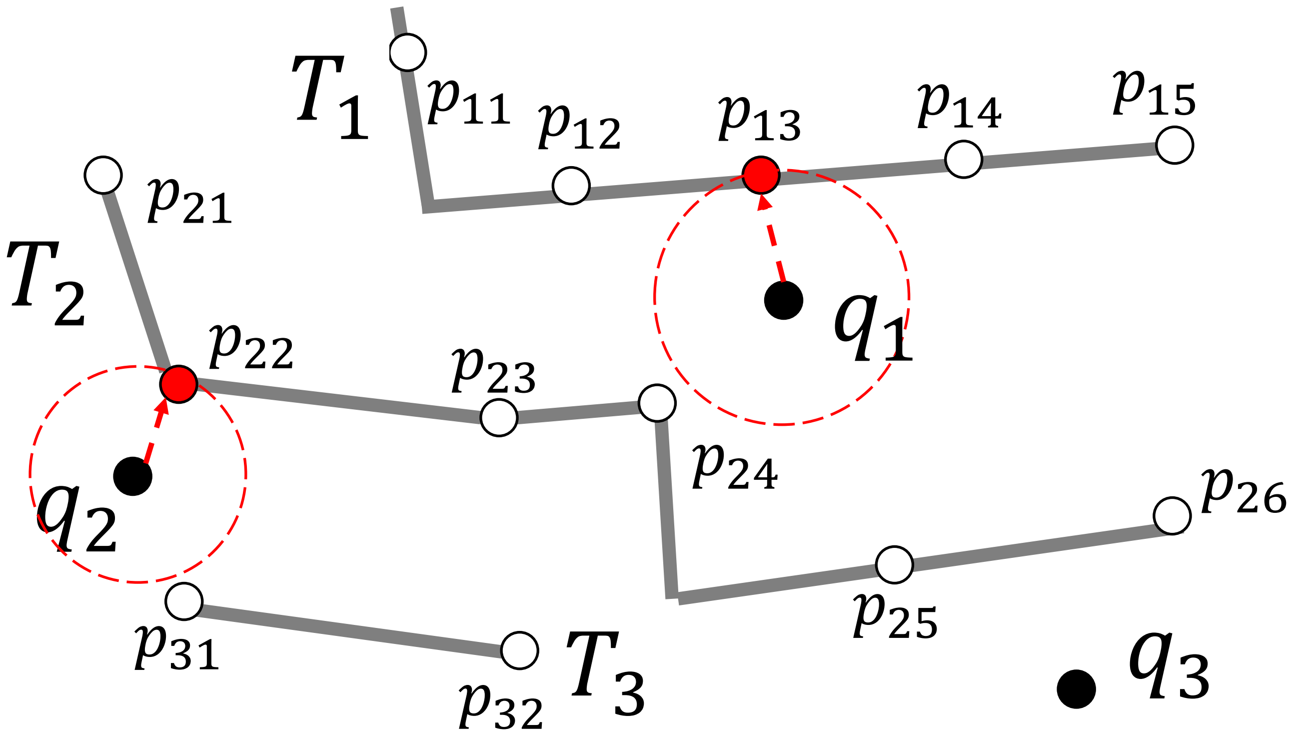
NN-Heaps



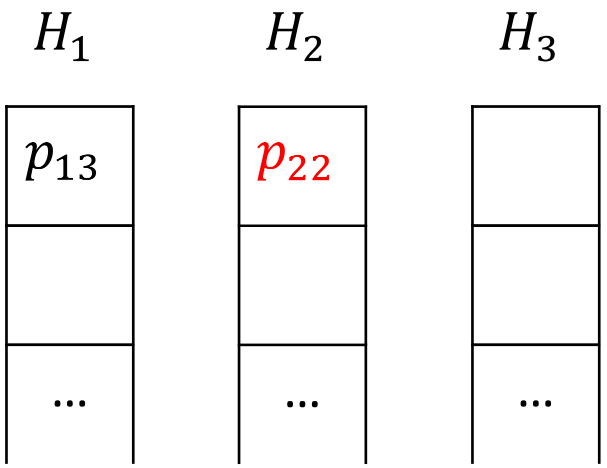
Candidates

	q_1	q_2	q_3
T_1	p_{13}	/	/

Generation with NN queries



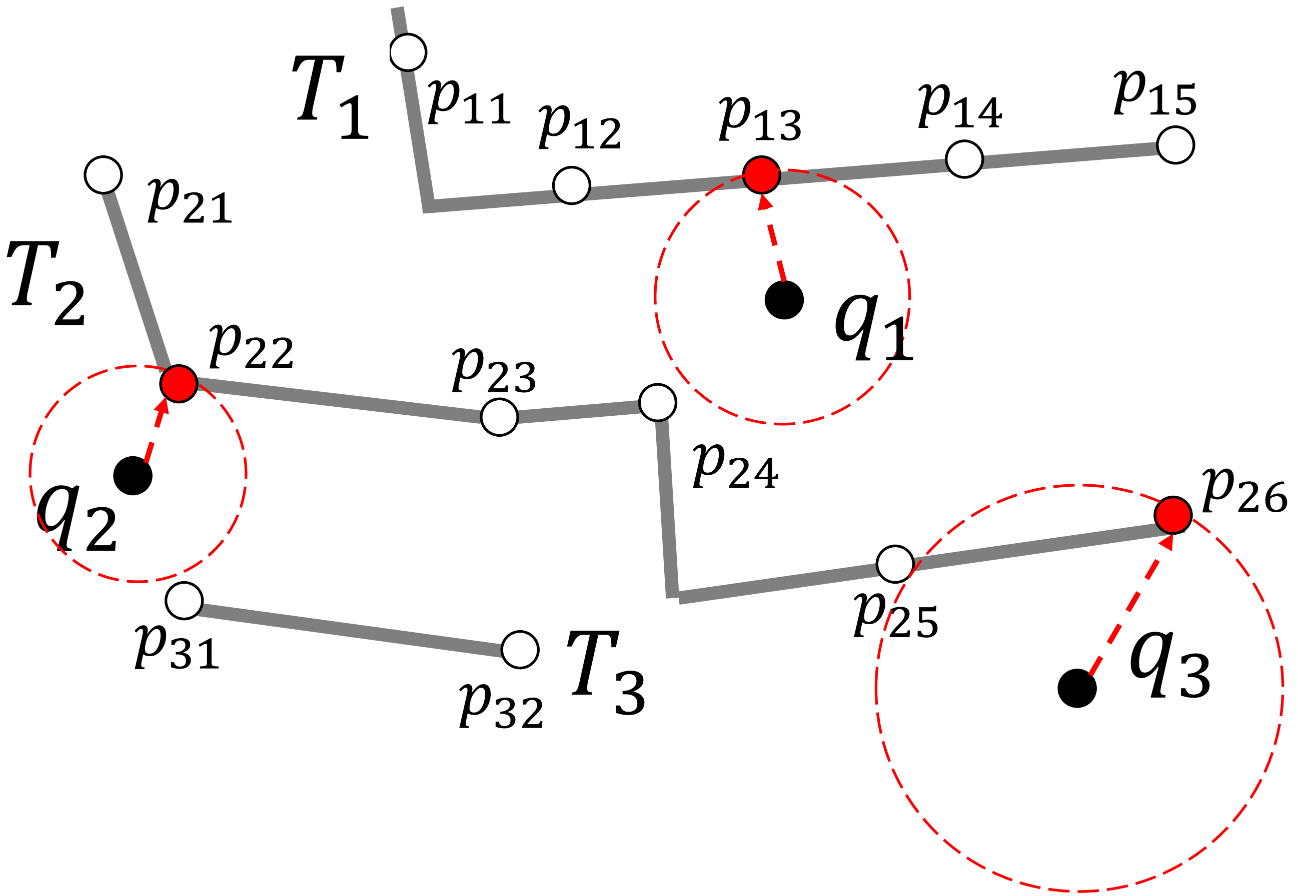
NN-Heaps



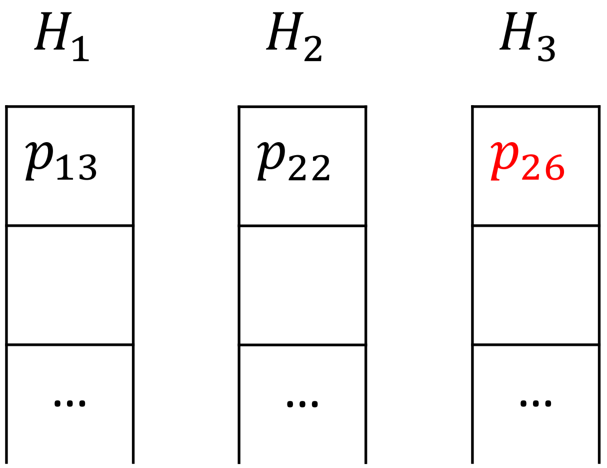
Candidates

	q_1	q_2	q_3
T_1	p_{13}	/	/
T_2	/	p_{22}	/

Generation with NN queries



NN-Heaps



Candidates

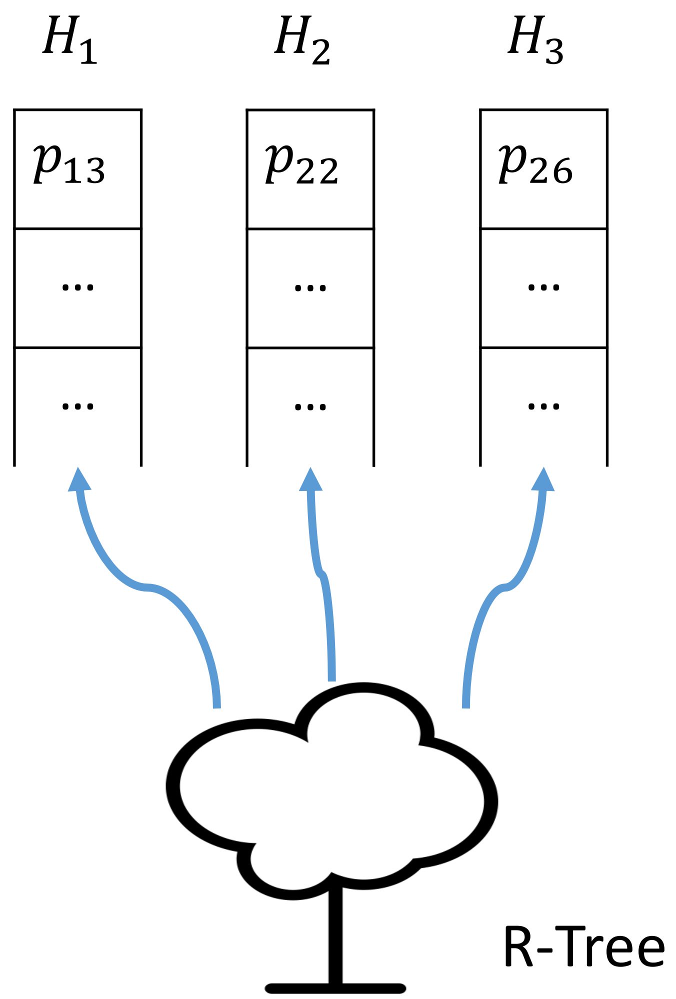
	q_1	q_2	q_3
T_1	p_{13}	/	/
T_2	/	p_{22}	p_{26}

Termination and Refinement (Trivial)

- $UB_k = k$th worst bound of existing candidates
- $LB =$ bound of unseen trajectories
- When $UB_k \leq LB$, refine the candidates and select the top-k as results

Drawbacks of NN-Based Solutions

- ***Independent*** NN-queries for each $q_i \in Q$
- **<u>I/O</u>**: R-tree nodes and trajectory points may be accessed multiple times
- **<u>CPU</u>**: NN heaps are costly to maintain

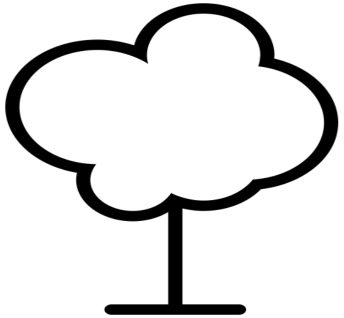


What we thought

- Can we access the R-tree nodes at most once?
- Can we avoid costly heap maintenance?
- Can we hold intermediate entries in one single structure?

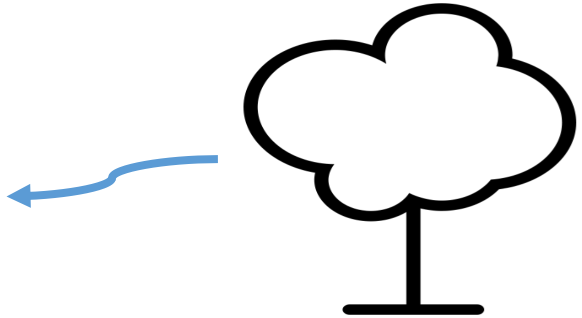
Spatial Range-Based Approach (SRA)

- Initialize list N = {root of R-Tree}
- For each $q_i \in Q$, do range query with radius r_i on N
  - N will be expanded during range queries
- Use accessed trajectory points to update existing candidates
- Go to refine when termination condition satisfies, otherwise increase r_i and repeat



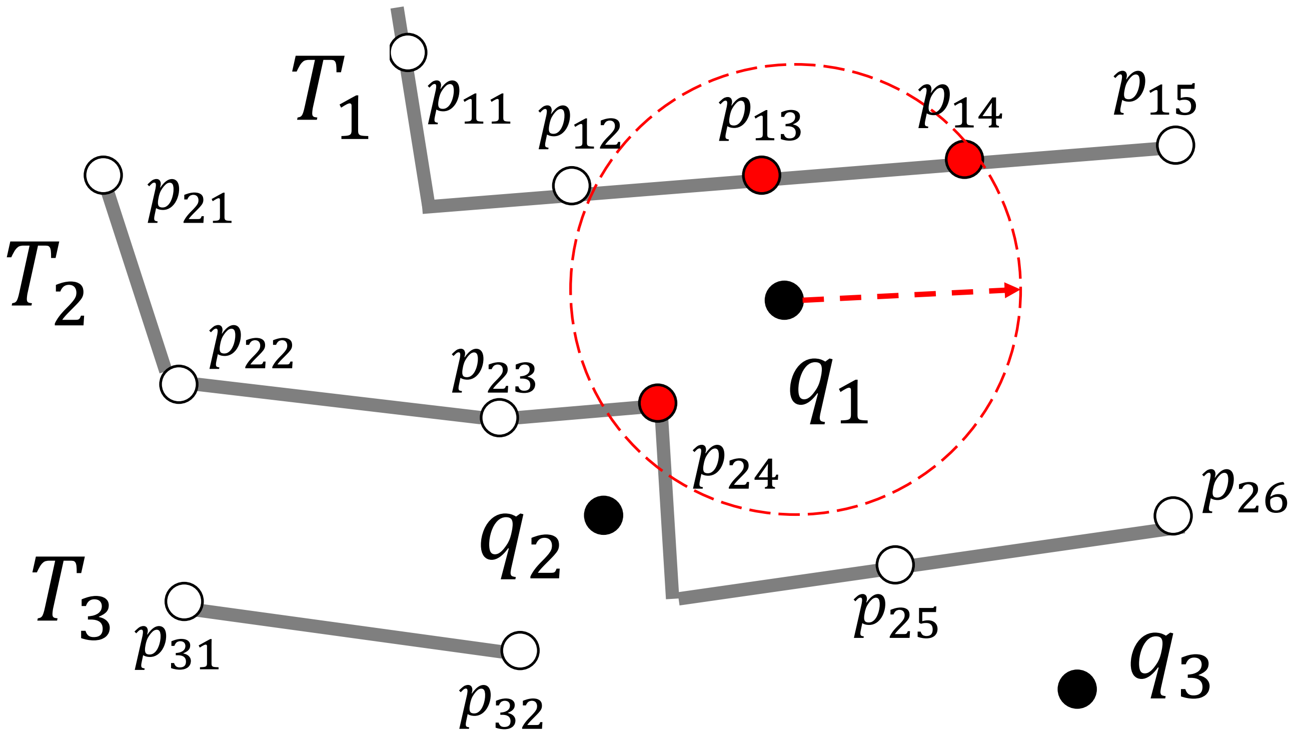
Spatial Range-Based Approach (SRA)

- Initialize list N = {root of R-Tree}
- For each $q_i \in Q$, do range query with radius r_i on N
  - N will be expanded during range queries



- Use accessed trajectory points to update existing candidates
- Go to refine when termination condition satisfies, otherwise increase r_i and repeat
- Challenges
  - Correctness
  - Node pruning
  - Termination condition

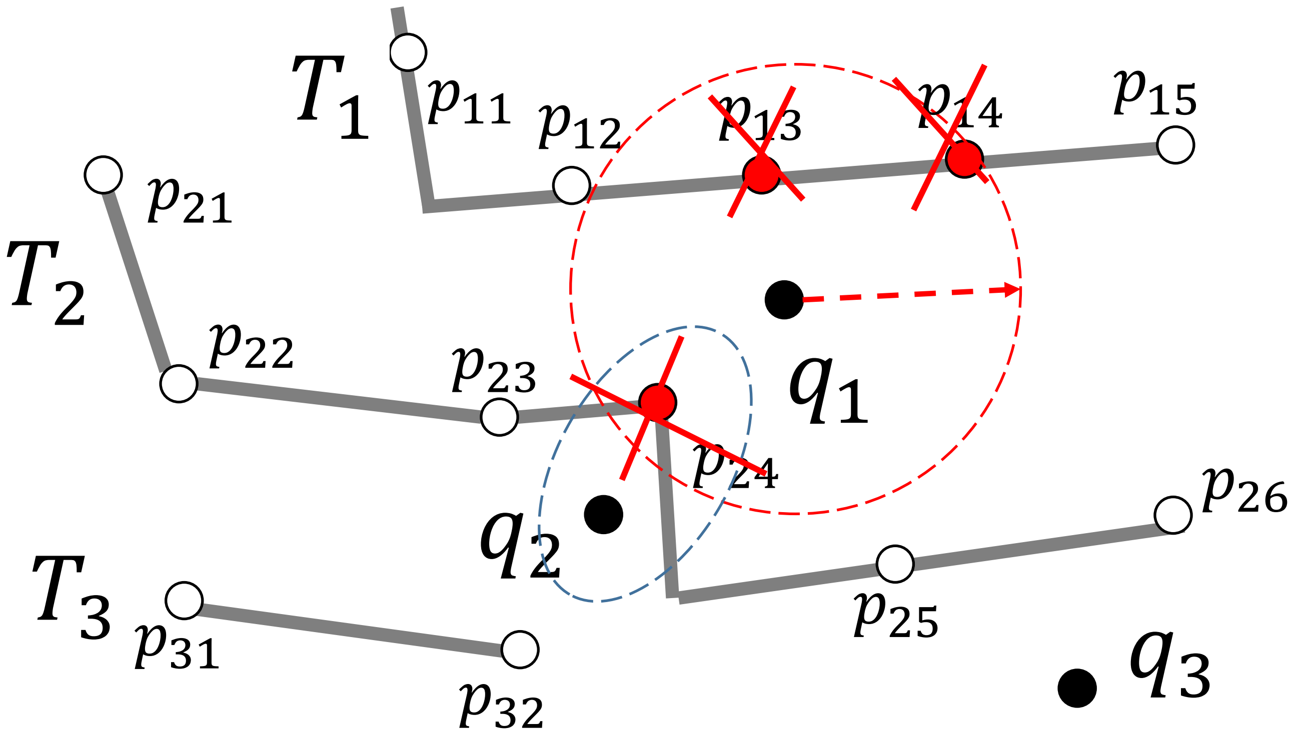
Correctness



- $\{p_{13}, p_{14}, p_{24}\}$ are accessed and **removed from N**

	q_1	q_2	q_3
T_1	p_{13}	/	/
T_2	p_{24}	/	/

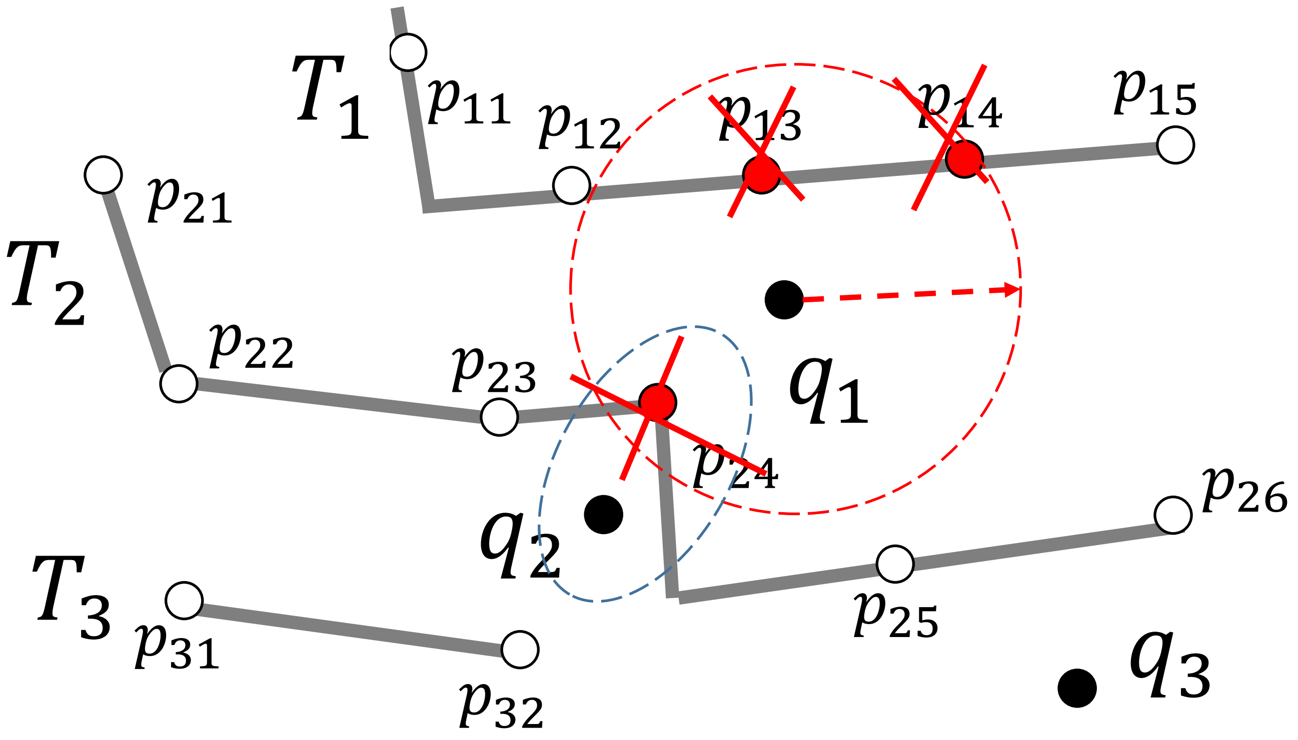
Correctness



- p_{24} will NOT be accessed by q_2, but it is actually the *matching location* for q_2

	q_1	q_2	q_3
T_1	p_{13}	/	/
T_2	p_{24}	/	/

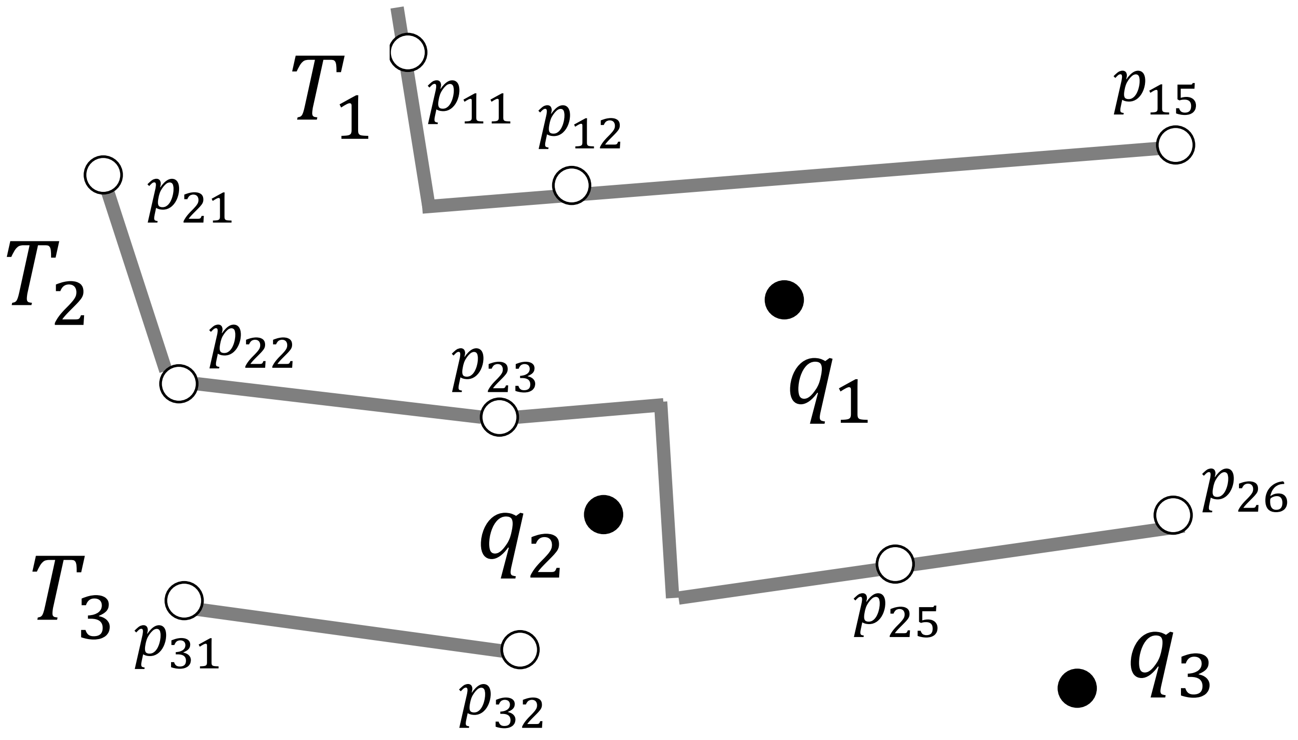
Temporary Matching Point



- $\{p_{13}, p_{14}, p_{24}\}$ are considered as temporary matching points for q_2 and q_3

	q_1	q_2	q_3
T_1	p_{13}	$*\ p_{13}$	$*\ p_{14}$
T_2	p_{24}	$*\ p_{24}$	$*\ p_{24}$

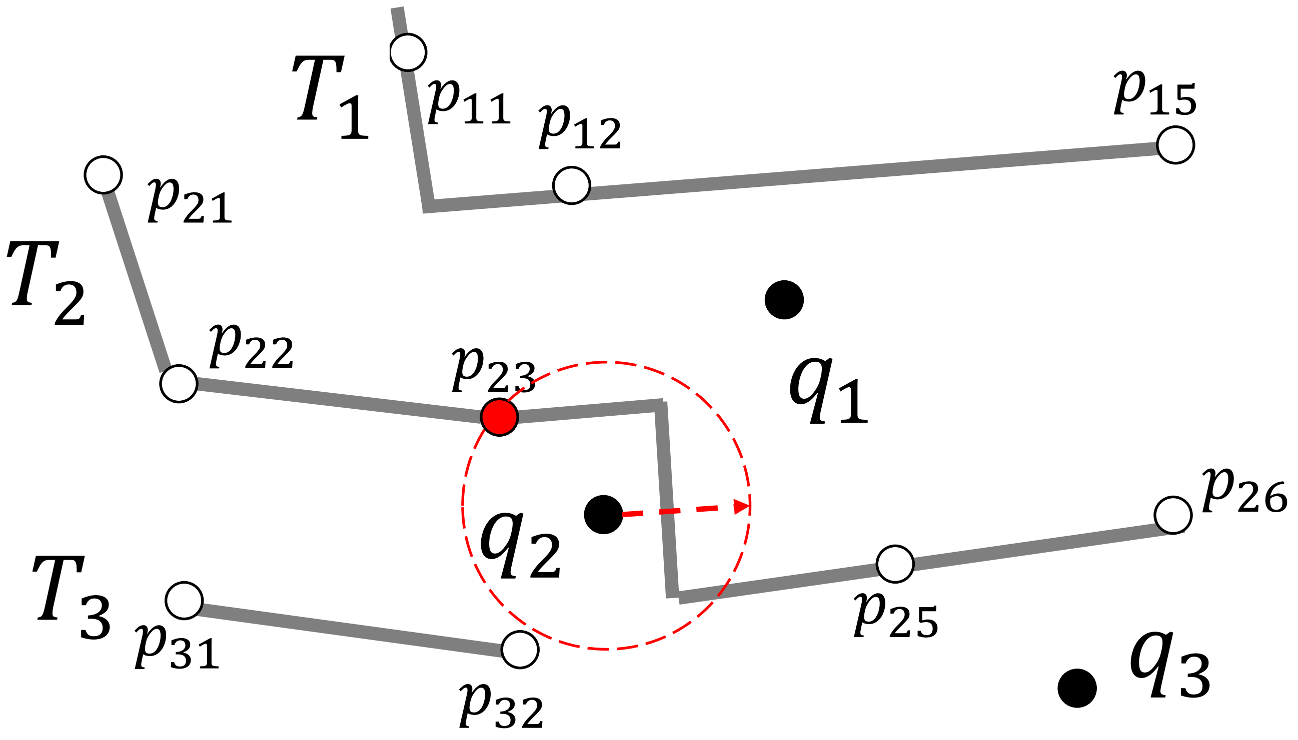
Temporary Matching Point



- $\{p_{13}, p_{14}, p_{24}\}$ can be safely removed

	q_1	q_2	q_3
T_1	p_{13}	$* p_{13}$	$* p_{14}$
T_2	p_{24}	$* p_{24}$	$* p_{24}$

Temporary Matching Point



- q_2 cannot find better match in T_2 than p_{24}
- p_{24} is finalized at (q_2, T_2)

	q_1	q_2	q_3
T_1	p_{13}	$* \, p_{13}$	$* \, p_{14}$
T_2	p_{24}	p_{24}	$* \, p_{24}$

Entry pruning

	q_1	q_2	q_3	UB
T_1	p_{13}	$* p_{13}$	$* p_{14}$	θ_1
T_2	p_{24}	p_{24}	$* p_{24}$	θ_2

$\theta = k$th minimum UB

e.g., Suppose $k = 1$
$\theta = \min(\theta_1, \theta_2)$

Entry pruning

	q_1	q_2	q_3	UB
T_1	p_{13}	$* p_{13}$	$* p_{14}$	θ_1
T_2	p_{24}	p_{24}	$* p_{24}$	θ_2

$\theta = k$th minimum UB

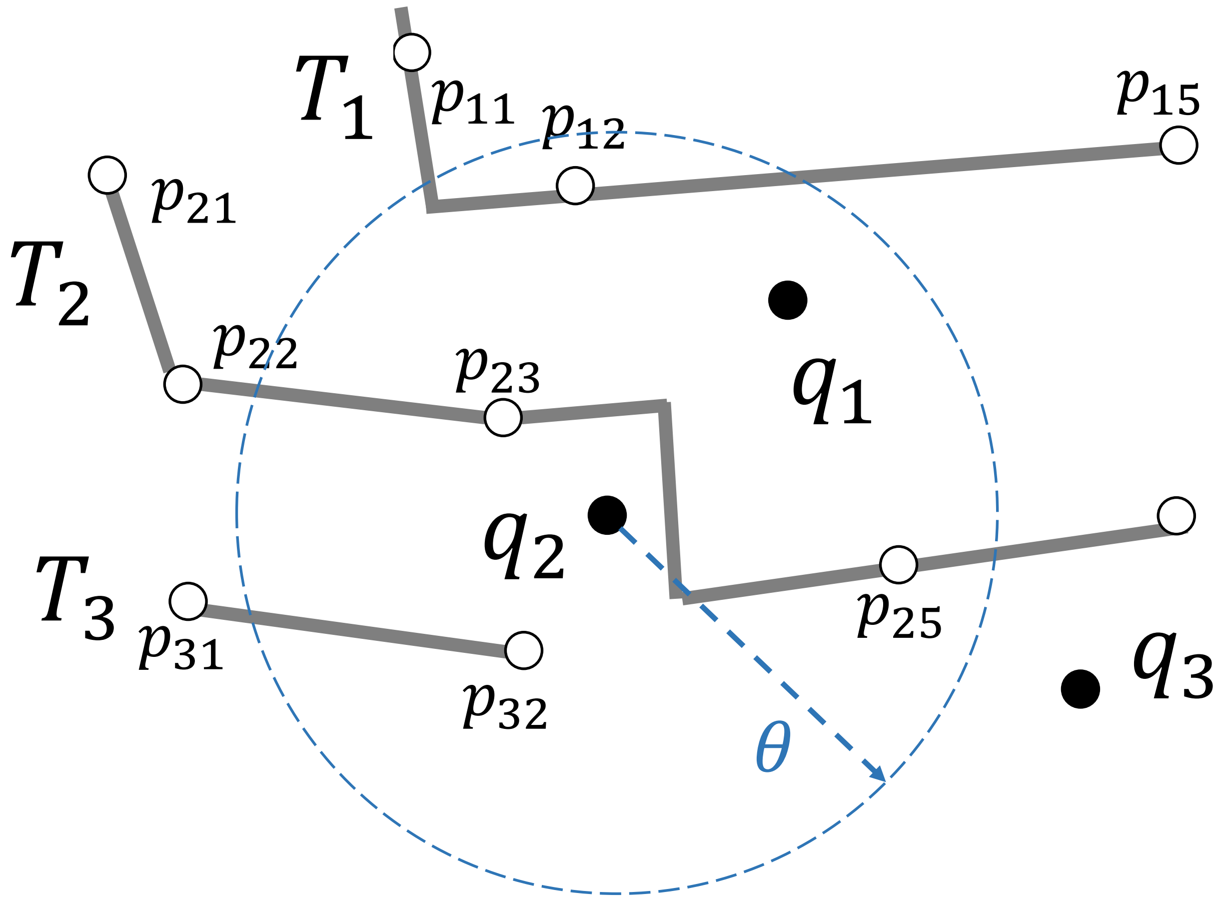
e.g., Suppose $k = 1$
$\theta = \min(\theta_1, \theta_2)$

Lemma:
For an unseen trajectory T_i,
it might become top-*k* only if
$dist(T_i, Q) \leq \theta$

Radius Bounding

Lemma:
For an unseen trajectory T_i, it might become top-*k* only if $dist(T_i, Q) \leq \theta$

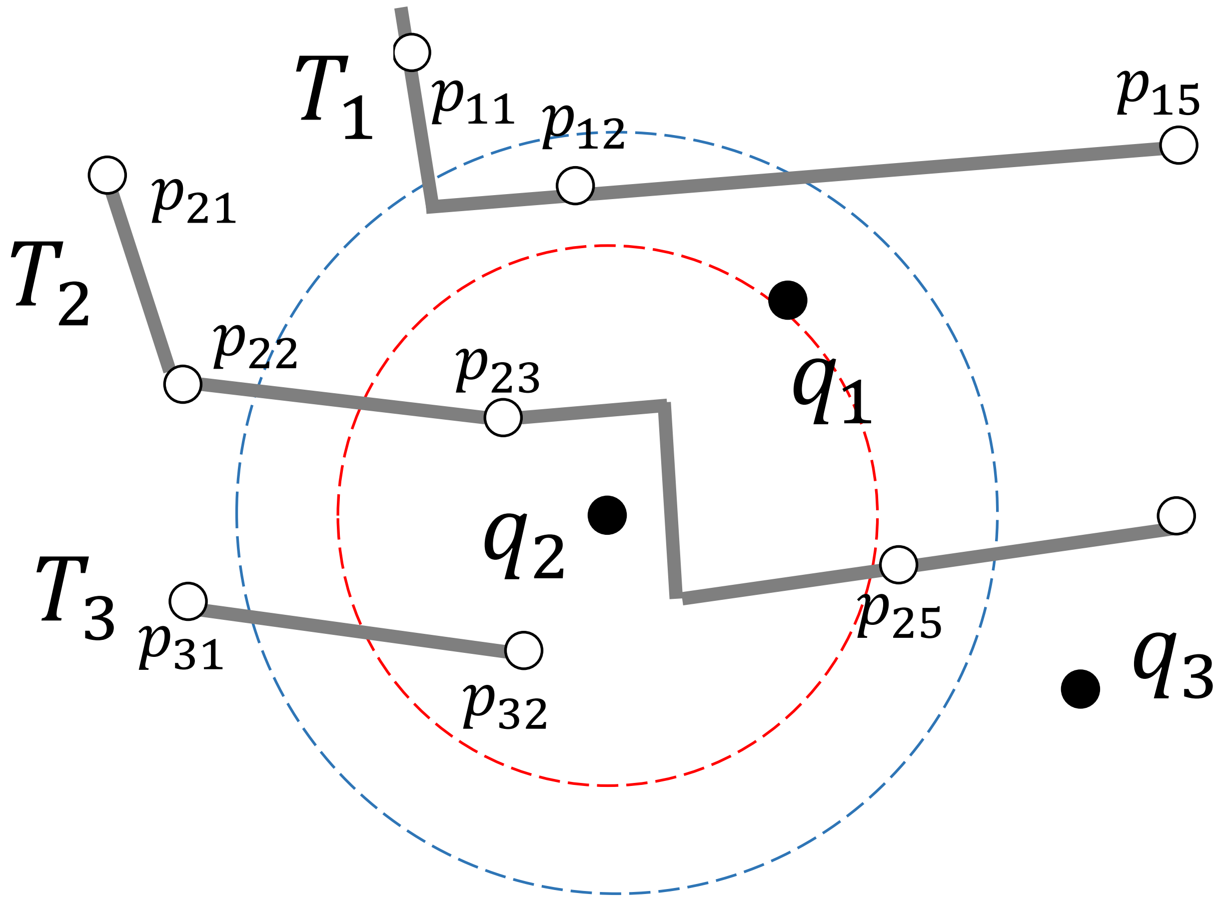
Radius Bounding: Any entry outside θ can be pruned. *



*Bound can be further tightened, details in the paper

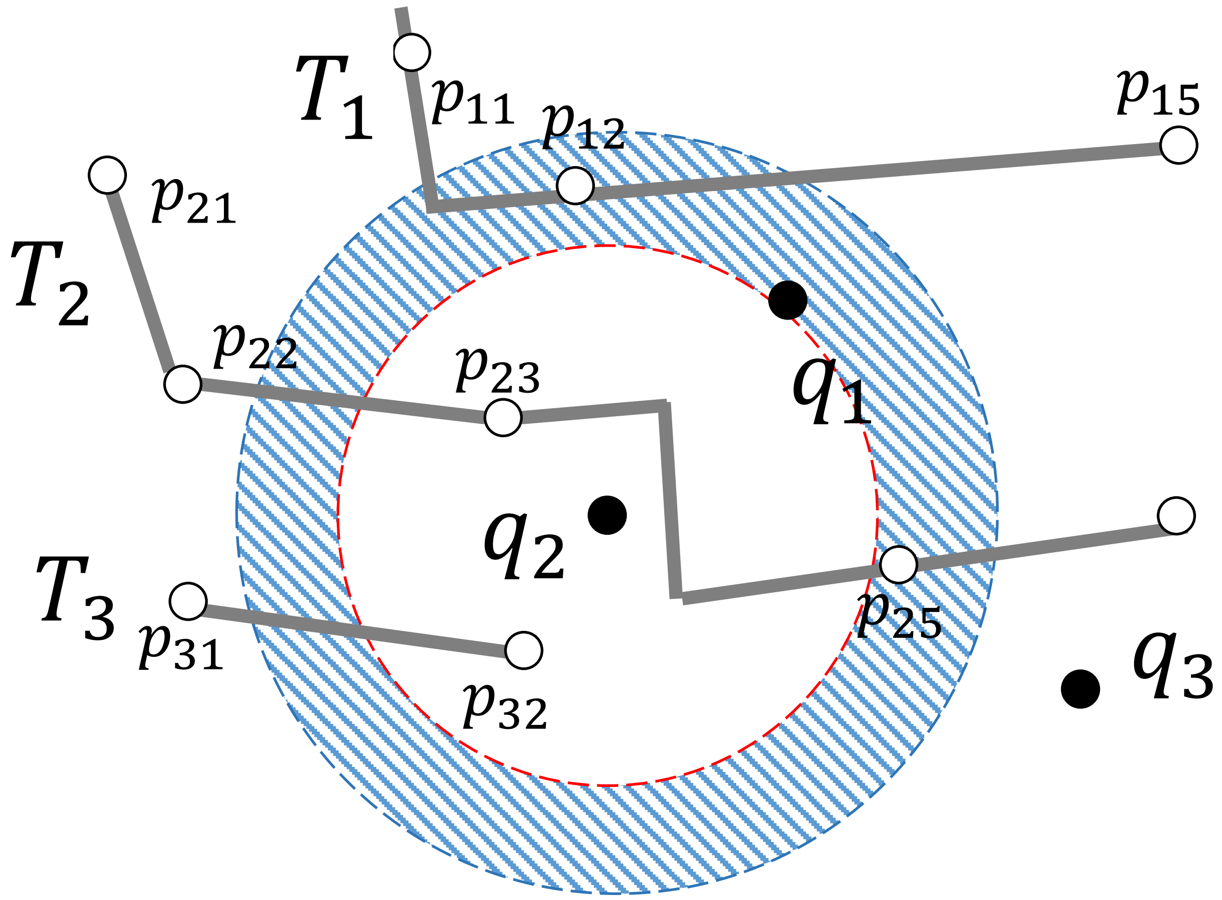
Termination Condition

- r_i increases
- θ decreases



Termination Condition

- r_i increases
- θ decreases
- If $r_i \geq \theta$, the candidate generation can be safely terminated



SRA+

- The idea of QE can be used in SRA as well
- During candidate generation, fill some partial matches

	q_1	q_2	q_3	UB
T_1	p_{13}	$* \; p_{13}$	$* \; p_{14}$	θ_1
T_2	p_{24}	p_{24}	$* \; p_{24}$	θ_2

SRA+

- The idea of QE can be used in SRA as well
- During candidate generation, fill some partial matches
- Decrease θ faster

	q_1	q_2	q_3	UB
T_1	p_{13}	$* p_{13}$	$* p_{14}$	θ_1
T_2	p_{24}	p_{24}	p_{26}	$\theta_2{}'$

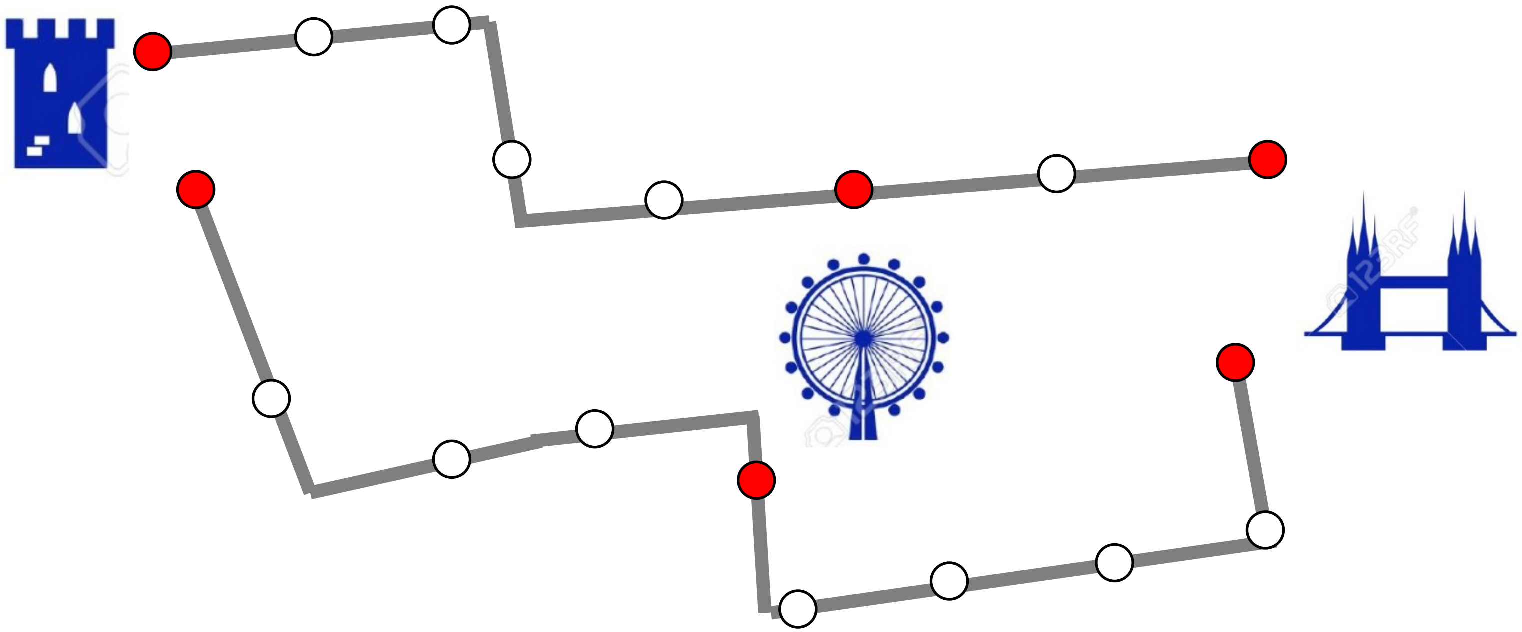
SRA/SRA+

- Keep entries in unordered list N
- Access each node/data at most once
- Shorten N with radius bounding
- Terminate when radiuses match
- Good applicability in other variants

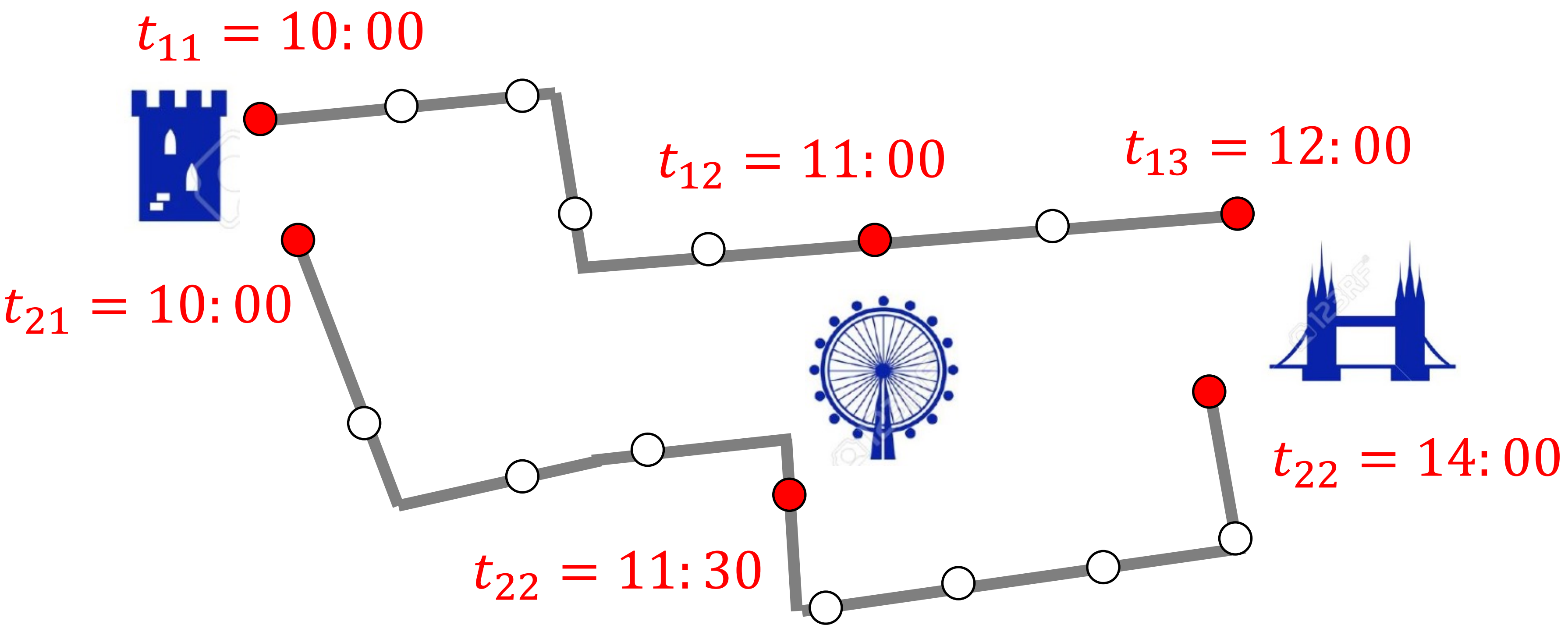
Outline

- Distance-Based Trajectory Search
  - Existing solutions and their drawbacks
  - SRA and SRA+: Novel and efficient approaches
- Bounded Distance-Based Trajectory Search
- Other variants
- Experiments
- Conclusion

Bounded Distance-Based Trajectory Search



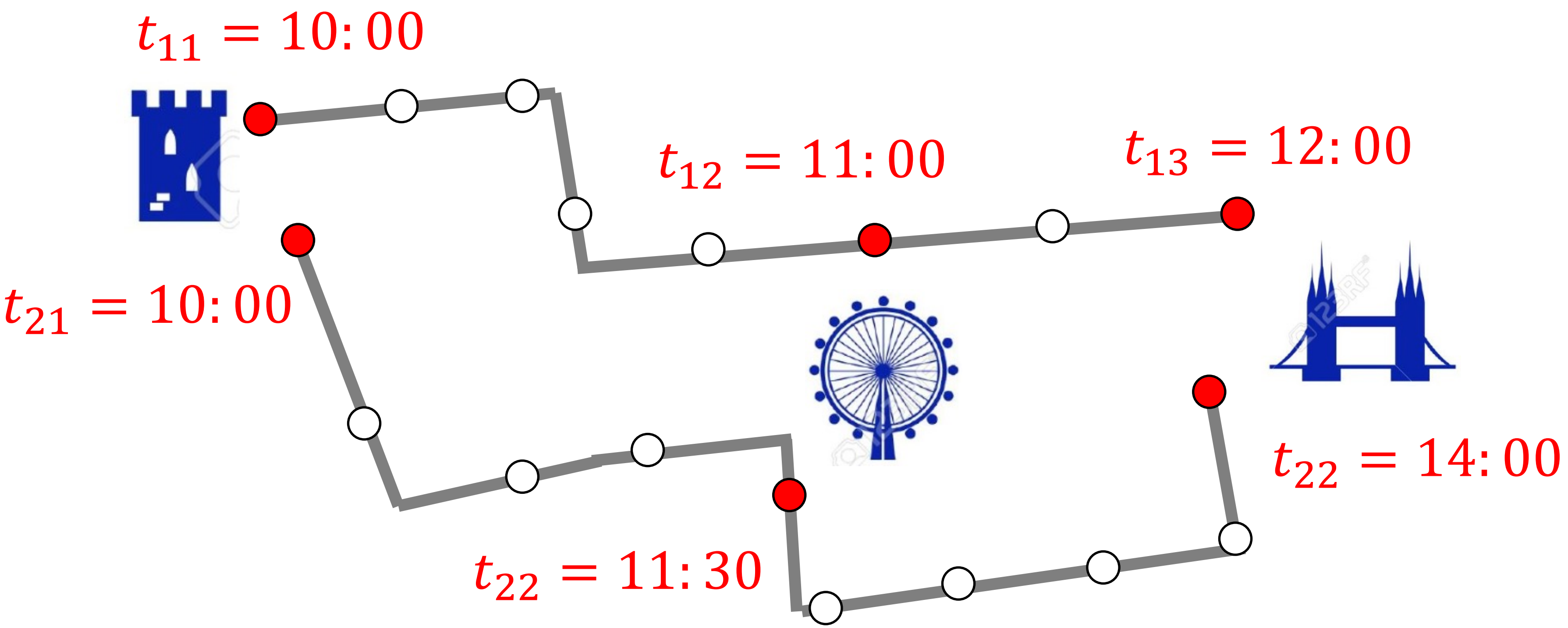
Bounded Distance-Based Trajectory Search



$$Span(T_1, Q) = 2\ h$$

$$Span(T_2, Q) = 4\ h$$

Bounded Distance-Based Trajectory Search



$Span(T_1, Q) = 2\ h$

$\cancel{Span(T_2, Q) = 4\ h}$

Bounded Distance-Based Trajectory Search (k-BDTS)

- Given a trajectory database T, a set of locations $Q = \{q_1, q_2, \dots, q_n\}$, and a span threshold τ, *k*-BDTS finds the top-*k* closest trajectories to Q with **span** $\leq \boldsymbol{\tau}$

k-BDTS: Incremental Algorithm

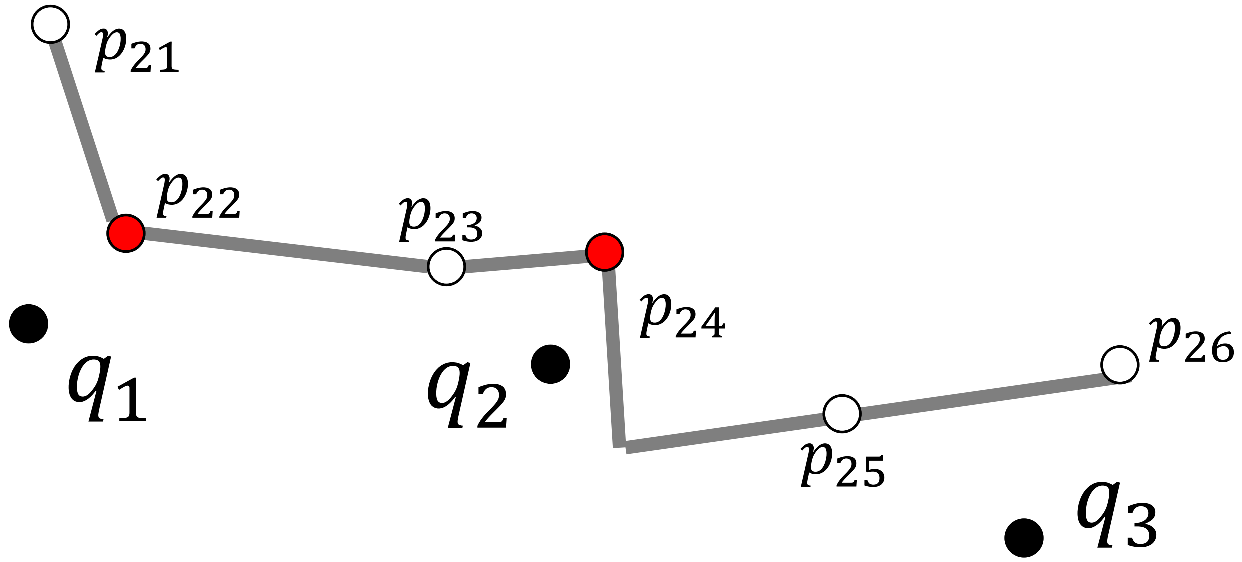
- (1) Result set $R = \emptyset$
- (2) Next k candidates R' = k-DTS (T, Q)
- (3) Update R with R', using span threshold τ
- (4) If $|R| \geq k$, return R
- (5) Otherwise go back to (1)

k-BDTS: Incremental Algorithm

- (1) Result set $R = \emptyset$
- (2) Next k candidates R' = k-DTS (T, Q)
- (3) Update R with R', using span threshold τ
- (4) If $|R| \geq k$, return R
- (5) Otherwise go back to (1)

- DTS has no idea whether a trajectory is valid or not (w.r.t. span)
- It may go through several rounds of generation-refinements

k-BDTS: One-Pass



	q_1	q_2	q_3	$span$
T_2	p_{22}	p_{24}	/	$\geq \lvert p_{24}.t - p_{22}.t \rvert$

- Follow DTS generation-refinement procedure
- Prune a candidate with span lower bound $> \tau$
- Lemma: When there are at least k fully matched candidates with span $\leq \tau$, calculate θ accordingly

Other Variants

- Distance and Span-based Trajectory Search (k-DSTS)
  - Instead of binary selection based on span, rank the trajectories based on a **combined distance and span score**
  - $f(T_i, Q) = \alpha \times dist(T_i, Q) + (1 - \alpha) \times span(T_i, Q)$
  - Adapt by replacing distance function and upper/lower bound calculation
- Order-aware Trajectory Search
  - Q is not a set but a sequence with **order constraint**
  - Distance function defined recursively taking the order into account
  - Upper/lower bounds adjusted accordingly

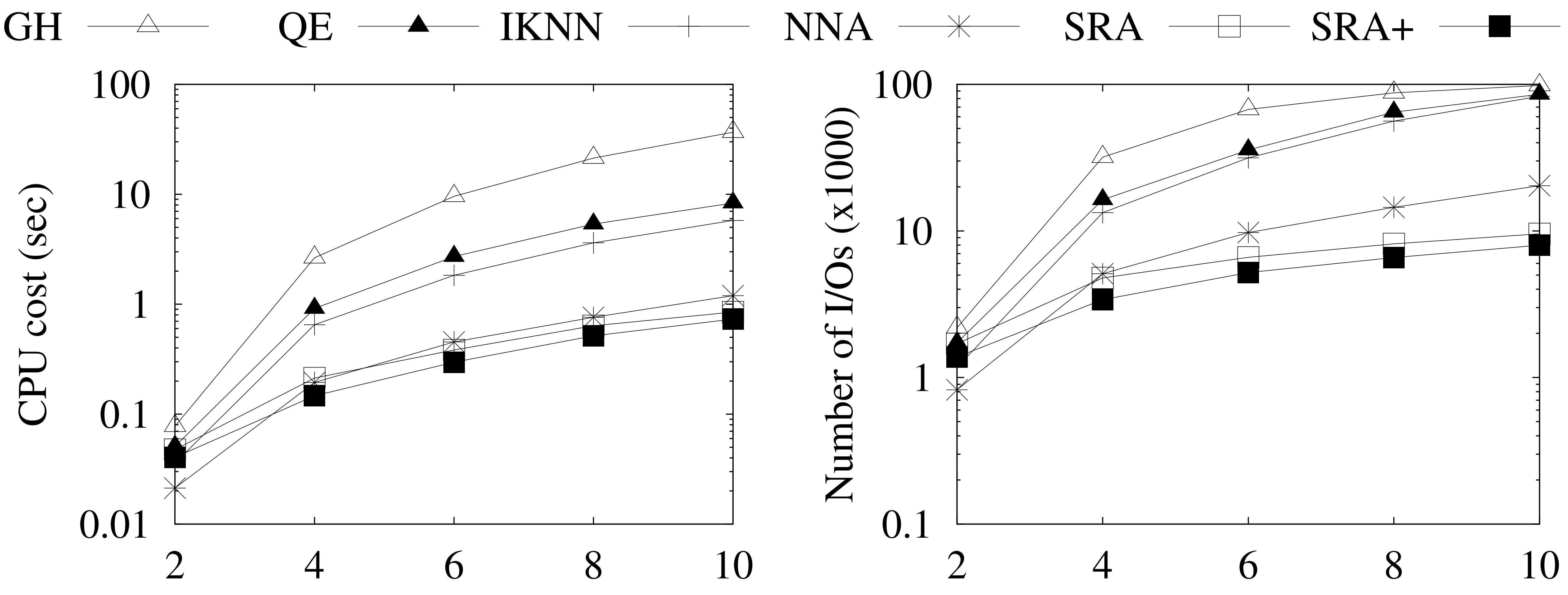
Outline

- Distance-Based Trajectory Search
  - Existing solutions and their drawbacks
  - SRA and SRA+: Novel and efficient approaches
- Bounded Distance-Based Trajectory Search
- Other variants
- Experiments
- Conclusion

Experiments

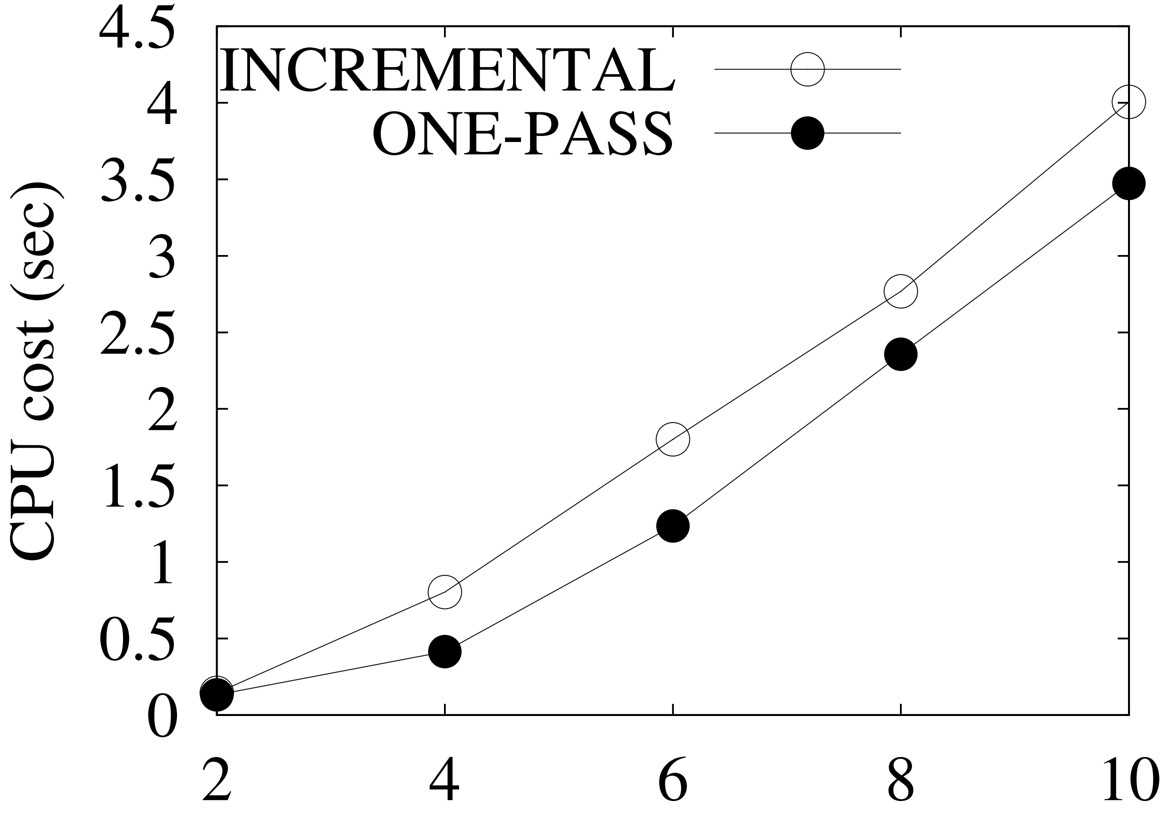
- Trajectory Dataset: GeoLife Project [Zheng 2009]
  - 17K trajectories with 19M points
  - Indexed by R-tree
- Query Locations: Randomly selected from different categories of 90K POIs in Beijing
  - $|Q| = \{2, 4, 6, 8, 10\}$
- Vary k, $|Q|$, τ, etc.

k-DTS: Varying $|Q|$

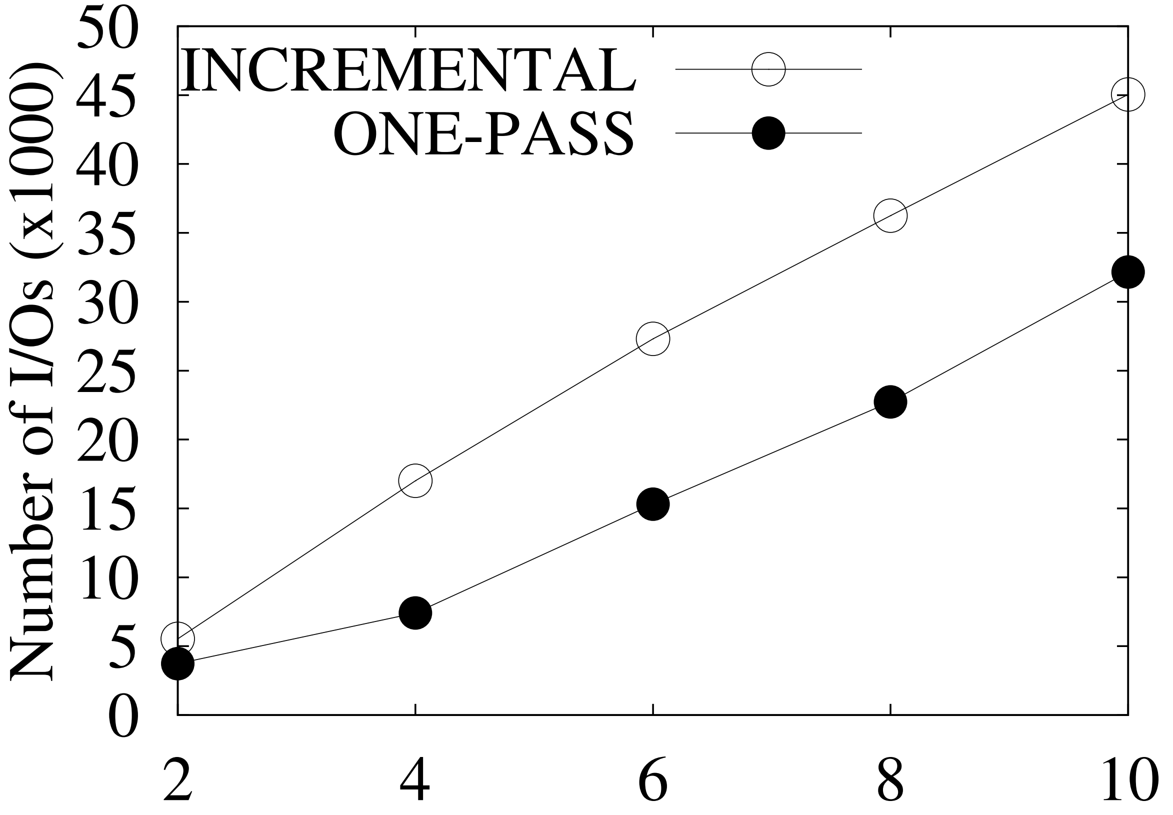


CPU Cost **Number of I/Os**

k-BDTS: Varying $|Q|$

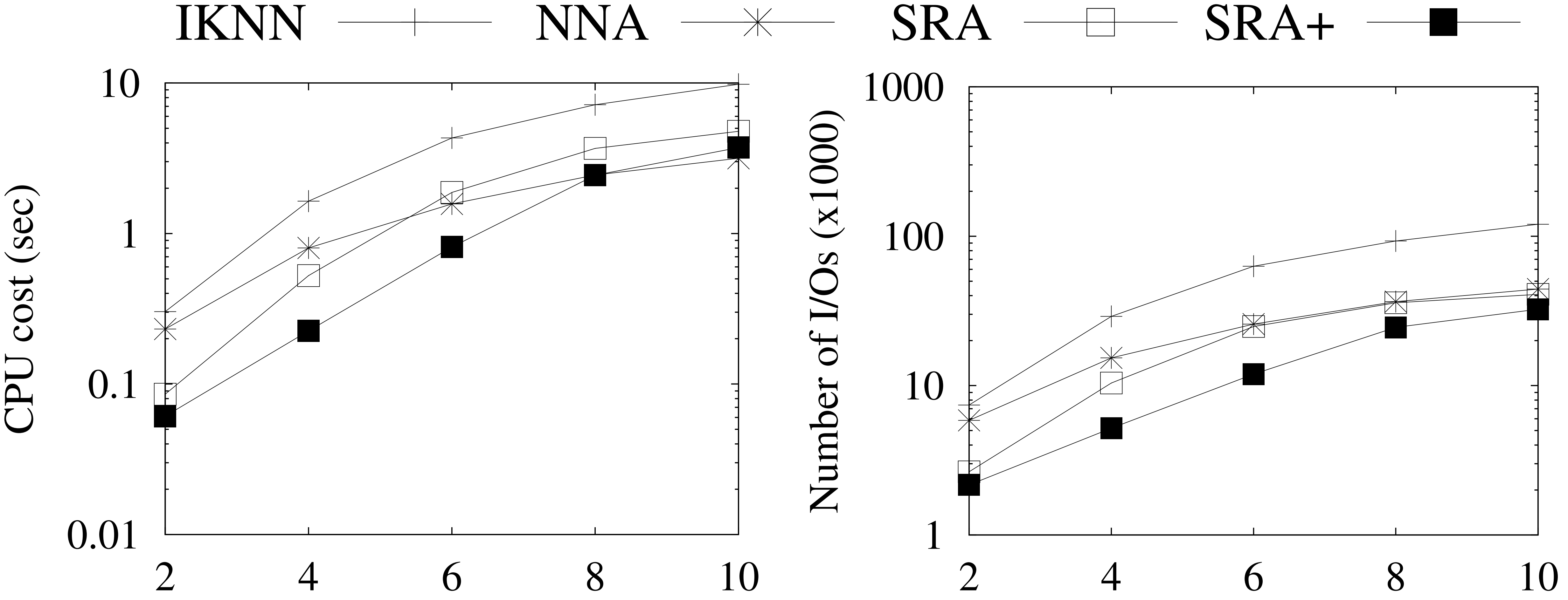


CPU Cost



Number of I/Os

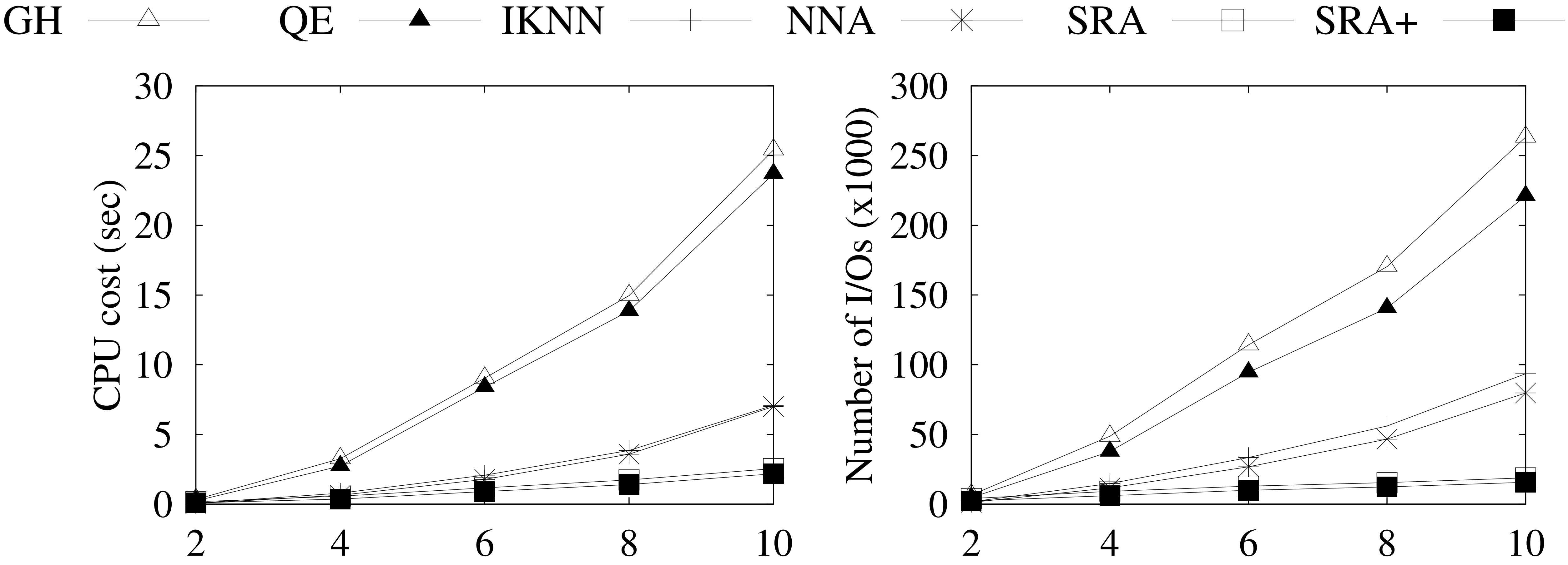
k-DSTS: Varying $|Q|$



CPU Cost

Number of I/Os

Ordered DTS: Varying $|Q|$



CPU Cost

Number of I/Os

Conclusions and Future Work

- SRA/SRA+: efficient and robust solutions to point-based trajectory search
- Applicability
  - Distance-based Trajectory Search
  - Bounded DTS
  - Ordered DTS
  - Distance & Span-based Trajectory Search
- Current/Future Work
  - Usage of SRA in more trajectory retrieval tasks
  - More general trajectory retrieval mechanism, e.g., set-based trajectory search

References

- [Chen 2010] Chen, Z., Shen, H.T., Zhou, X., Zheng, Y., Xie, X.: Searching trajectories by locations: an efficiency study. In: SIGMOD. (2010)
- [Tang 2011] Tang, L.A., Zheng, Y., Xie, X., Yuan, J., Yu, X., Han, J.: Retrieving k-nearest neighboring trajectories by a set of point locations. In: SSTD. (2011)
- [Zheng 2009] Zheng, Y., Zhang, L., Xie, X., Ma, W. Y.: Mining interesting locations and travel sequences from GPS trajectories. In: WWW. (2009)

Thanks!

Q & A